Exhibit 20

Sonos

The sonos integration allows you to control your Sonos wireless speakers from Home Assistant. It also works with IKEA Symfonisk speakers.

Configuration

Adding Sonos to your Home Assistant instance can be done via the user interface, by using this My button:

Sonos can be auto-discovered by Home Assistant. If an instance was found, it will be shown as *"Discovered"*, which you can select to set it up right away.

Manual configuration steps

Feature controls & sensors

Speaker-level controls are exposed as number or switch entities. Additionally, various sensor and binary_sensor entities are provided.

Controllable features

- **All devices**: Alarms, Bass, Treble, Loudness, Crossfade, Status Light, Touch Controls
- **Home theater devices**: Audio Delay ("Lip Sync"), Night Sound, Speech Enhancement, Surround Enabled, Surround Music Full Volume ("Full/Ambient"), Surround Level ("TV Level"), Music Surround Level
- **When paired with a sub**: Subwoofer Enabled, Subwoofer Gain

Sensors

- **Each Sonos system**: Sonos Favorites
- **Devices with battery**: Battery level, Power state
- **Home theater devices**: Audio Input Format
- **Voice-enabled devices**: Microphone Enabled

Battery support notes

Battery sensors are fully supported for the Sonos Roam and Sonos Move devices on S2 firmware. Sonos Move speakers still on S1 firmware are supported but may update infrequently.

For each speaker with a battery, a sensor showing the current battery charge level and a binary_sensor showing the power state of the speaker are created. The binary_sensor reports if the speaker is currently powered by an external source and its power_source attribute shows which specific source is providing the current power. This source attribute can be one of BATTERY, SONOS_CHARGING_RING if using wireless charging, or USB_POWER if charging via USB cable. Note that the Roam will report SONOS_CHARGING_RING even when using a generic Qi charger.

The battery sensors rely on working change events or updates will be delayed. S1 battery sensors **require** working events to report any data. See more details in Advanced use.

Alarm support notes

The Sonos integration adds one `switch` for each alarm set in the Sonos app. The alarm switches are detected, deleted and assigned automatically and come with several attributes that help to monitor Sonos alarms.

Microphone support notes

The microphone can only be enabled/disabled from physical buttons on the Sonos device and cannot be controlled from Home Assistant. A `binary_sensor` reports its current state.

Sonos Favorites support notes

The favorites sensor provides the names and `media_content_id` values for each of the favorites saved to My Sonos in the native Sonos app. This sensor is intended for users that need to access the favorites in a custom template. For most users, accessing favorites by using the Media Browser functionality and "Play media" script/automation action is recommended.

If using the provided `media_content_id` with the `media_player.play_media` service, the `media_content_type` must be set to "favorite_item_id".

Example templates:

```
# Get all favorite names as a list (old behavior)
{{ state_attr("sensor.sonos_favorites", "items").values() | list }}

# Pick a specific favorite name by position
{{ (state_attr("sensor.sonos_favorites", "items").values() | list)[3] }}

# Pick a random item's `media_content_id`
{{ state_attr("sensor.sonos_favorites", "items") | list | random }}

# Loop through and grab name & media_content_id
{% for media_id, name in state_attr("sensor.sonos_favorites", "items").items() %}
  {{ name, media_id }}
{% endfor %}
```

YAML

The Sonos favorites sensor (`sensor.sonos_favorites`) is disabled by default. It can be found and enabled from the entities associated with the Sonos integration on your Devices & Services page.

Playing media

Sonos accepts a variety of `media_content_id` formats in the `media_player.play_media` service, but most commonly as URIs. For example, both Spotify and Tidal share links can be provided as-is. Playback of music hosted on a Plex server is possible. Direct HTTP/HTTPS links to local or remote media files can also be used if the Sonos device can reach the URI directly, but specific media encoding support may vary.

Music services which require an account (e.g., Spotify) must first be configured using the Sonos app.

An optional `enqueue` argument can be added to the service call. If `true`, the media will be appended to the end of the playback queue. If not provided or `false` then the queue will be replaced.

Examples:

This is an example service call that plays an audio file from a web server on the local network (like the Home Assistant built-in webserver):

```
service: media_player.play_media
target:
  entity_id: media_player.sonos
data:
  media_content_type: "music"
  media_content_id: "http://192.168.1.50:8123/local/sound_files/doorbell-front.mp3"
```

YAML

Sonos can also play music or playlists from Spotify. Both Spotify URIs and URLs can be used directly. An example service call using a playlist URI:

```
service: media_player.play_media
target:
```

```
  entity_id: media_player.sonos
data:
  media_content_type: "playlist"
  media_content_id: "spotify:playlist:abcdefghij0123456789XY"
  enqueue: true
```
YAML

An example service call using a Spotify URL:
```
service: media_player.play_media
target:
  entity_id: media_player.sonos
data:
  media_content_type: "music"
  media_content_id: "https://open.spotify.com/album/abcdefghij0123456789YZ"
```
YAML

Run a Plex Media Server in your home? The Sonos integration can work with that as well. This example plays music directly from your Plex server:
```
service: media_player.play_media
target:
  entity_id: media_player.sonos
data:
  media_content_type: "music"
  media_content_id: 'plex://{ "library_name": "Music", "artist_name": "M83", "album_name": "Hurry Up, We're Dreaming" }'
```
YAML

Services

The Sonos integration makes various custom services available in addition to the standard Media Player services.

Service sonos.snapshot

Take a snapshot of what is currently playing on one or more speakers. This service, and the following one, are useful if you want to play a doorbell or notification sound and resume playback afterwards.

The queue is not snapshotted and must be left untouched until the restore. Using `media_player.play_media` is safe and can be used to play a notification sound, including TTS announcements.

Service data attribute	Optional	Description
`entity_id`	yes	The speakers to snapshot. To target all Sonos devices, use `all`.
`with_group`	yes	Should we also snapshot the group layout and the state of other speakers in the group, defaults to true.

Service sonos.restore

Restore a previously taken snapshot of one or more speakers.

The playing queue is not snapshotted. Using `sonos.restore` on a speaker that has replaced its queue will restore the playing position, but in the new queue!
A cloud queue cannot be restarted. This includes queues started from within Spotify and queues controlled by Amazon Alexa.

Service data attribute	Optional	Description
`entity_id`	yes	String or list of `entity_ids` that should have their snapshot restored. To target all Sonos devices, use `all`.

Service data attribute	Optional	Description
`with_group`	yes	Should we also restore the group layout and the state of other speakers in the group, defaults to true.

Service sonos.set_sleep_timer

Sets a timer that will turn off a speaker by tapering the volume down to 0 after a certain amount of time. Protip: If you set the sleep_time value to 0, then the speaker will immediately start tapering the volume down.

Service data attribute	Optional	Description
`entity_id`	yes	String or list of `entity_id`s that will have their timers set.
`sleep_time`	no	Integer number of seconds that the speaker should wait until it starts tapering. Cannot exceed 86399 (one day).

Service sonos.clear_sleep_timer

Clear the sleep timer on a speaker, if one is set.

Service data attribute	Optional	Description
`entity_id`	no	String or list of `entity_id`s that will have their timers cleared. Must be a coordinator speaker.

Service sonos.update_alarm

Update an existing Sonos alarm.

Service data attribute	Optional	Description
`entity_id`	yes	String or list of `entity_id`s that will have their timers cleared. Must be a coordinator speaker.
`alarm_id`	no	Integer that is used in Sonos to refer to your alarm.
`time`	yes	Time to set the alarm.
`volume`	yes	Float for volume level.
`enabled`	yes	Boolean for whether or not to enable this alarm.
`include_linked_zones`	yes	Boolean that defines if the alarm also plays on grouped players.

Service sonos.play_queue

Starts playing the Sonos queue.

Force start playing the queue, allows switching from another stream (such as radio) to playing the queue.

Service data attribute	Optional	Description
`entity_id`	yes	String or list of `entity_id`s that will start playing. It must be the coordinator if targeting a group.
`queue_position`	yes	Position of the song in the queue to start playing from, starts at 0.

Service sonos.remove_from_queue

Removes an item from the queue.

Service data attribute	Optional	Description
`entity_id`	yes	String or list of `entity_id`s that will remove an item from the queue. It must be the coordinator if targeting a group.
`queue_position`	yes	Position in the queue to remove.

```
# Example automation to remove just played song from queue
alias: "Remove last played song from queue"
id: Remove last played song from queue
trigger:
  - platform: state
    entity_id: media_player.kitchen
  - platform: state
    entity_id: media_player.bathroom
  - platform: state
    entity_id: media_player.move
condition:
  condition: and
  conditions:
    # Coordinator
    - condition: template
      value_template: >
        {{ state_attr( trigger.entity_id , 'group_members')[0] ==  trigger.entity_id }}
    # Going from queue to queue
    - condition: template
      value_template: >
        {{ 'queue_position' in trigger.from_state.attributes and 'queue_position' in trigger.to_state.attributes }}
    # Moving forward
    - condition: template
      value_template: >
        {{ trigger.from_state.attributes.queue_position < trigger.to_state.attributes.queue_position }}
action:
  - service: sonos.remove_from_queue
    target:
      entity_id: >
        {{ trigger.entity_id }}
    data:
      queue_position: >
        {{ trigger.from_state.attributes.queue_position }}
YAML
```

Network requirements

To work optimally, the Sonos devices must be able to connect back to the Home Assistant host on TCP port 1400. This will allow the push-based updates to work properly. If this port is blocked or otherwise unreachable from the Sonos devices, the integration will fall back to a polling mode which is slower to update and much less efficient. The integration will alert the user if this problem is detected.

See Advanced use below for additional configuration options which may be needed to address this issue in setups with more complex network topologies.

Advanced use

For advanced uses, there are some manual configuration options available. These are usually only needed if you have a complex network setup where Home Assistant and Sonos are not on the same subnet.

You can disable auto-discovery by specifying the Sonos IP addresses:

```yaml
# Example configuration.yaml entry with manually specified Sonos IP addresses
sonos:
  media_player:
    hosts:
      - 192.0.2.25
      - 192.0.2.26
      - 192.0.2.27
```

YAML

If your Home Assistant instance has multiple IP addresses, you can select the specific IP address that should be used for Sonos auto-discovery with the Network integration. This should only be necessary if the Sonos speakers are on a network segment not reachable from the default interface.

The Sonos speakers will attempt to connect back to Home Assistant to deliver change events. By default, Home Assistant will listen on port 1400 but will try the next 100 ports above 1400 if it is in use. You can change the IP address that Home Assistant advertises to Sonos speakers. This can help in NAT scenarios such as when *not* using the Docker option `--net=host`:

```yaml
# Example configuration.yaml entry modifying the advertised host address
sonos:
  media_player:
    advertise_addr: 192.0.2.1
```

YAML